# 44장. REST API

- **REST(REpresentational State Transfer)**
  : HTTP의 장점을 최대한 활용할 수 있는 아키텍처
- **RESTful**
  : REST의 기본 원칙을 성실히 지킨 서비스 디자인

## 44.1 REST API의 구성

| 구성 요소 | 내용 | 표현 방법 |
|---|---|---|
| 자원(resource) | 자원 | URI(엔드포인트) |
| 행위(verb) | 자원에 대한 행위 | HTTP 요청 메서드 |
| 표현(representations) | 자원에 대한 행위의 구체적 내용 | 페이로드 |

## 44.2 REST API 설계 원칙

- REST에서 가장 중요한 기본적인 원칙 두 가지
  1. **URI는 리소스를 표현**하는 데 집중
  2. **행위에 대한 정의는 HTTP 요청 메서드**를 통해 하는 것

1. **URI는 리소스를 표현해야 한다.**
   - 이름은 동사보다는 명사를 사용, get 같은 행위에 대한 표현 사용 X

2. **리소스에 대한 행위는 HTTP 요청 메서드로 표현한다.**
   - 5가지 요청 메서드(GET, POST, PUT, PATCH, DELETE 등)을 사용하여 CRUD 구현

## 44.3 JSON Server를 이용한 REST API 실습

### 44.3.1 JSON Server 설치

1. JSON Server는 json 파일을 사용하여 가상 REST API 서버를 구축할 수 있는 툴
   ( npm을 사용하여 JSON Server를 설치 )

   > 💡 **npm(node package manager)**
   > : 자바스크립트 패키지 매니저

2. 터미널에 명령어 입력하여 JSON Server 설치

```
$ mkdir json-server-exam && cd json-server-exam
```

```
$ npm init -y
$ npm install json-server --save-dev
```

### 44.3.2 db.json 파일 생성

1. 프로젝트 루트 폴더(/json-server-exam)에 db.json 파일 생성
   - db.json 파일은 리소스를 제공하는 데이터베이스 역

### 44.3.3 JSON Server 실행

```
$ json-server –watch db.json
```

### 44.3.4 GET 요청

- get_index.html

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
    const xhr = new XMLHttpRequest();
    xhr.open('GET', '/todos');

    xhr.send();

    xhr.onload = () => {
      if (xhr.status === 200){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
      }
    };
  </script>
</body>
</html>
```

get_retrieve.html

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
```

```
    const xhr = new XMLHttpRequest();
    xhr.open('GET', '/todos/1');

    xhr.send();

    xhr.onload = () => {
      if (xhr.status === 200){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
      }
    };
  </script>
</body>
</html>
```

### 44.3.5 POST 요청

- post.html

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
    const xhr = new XMLHttpRequest();
    xhr.open('POST', '/todos');

    xhr.setRequestHeader('content-type', 'application/json');

    xhr.send(JSON.stringify({ id: 4, content: 'Angular', completed: false }))

    xhr.onload = () => {
      if ( xhr.status === 200 || xhr.status === 201 ){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
      }
    };
  </script>
</body>
</html>
```

### 44.3.6 PUT 요청

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
    const xhr = new XMLHttpRequest();
    xhr.open('PUT', '/todos/4');

    xhr.setRequestHeader('content-type', 'application/json');

    xhr.send(JSON.stringify({ id: 4, content: 'React', completed: true }));

    xhr.onload = () => {
      if ( xhr.status === 200 ){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
      }
    };
  </script>
</body>
</html>
```

### 44.3.7 PATCH 요청

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
    const xhr = new XMLHttpRequest();
    xhr.open('PATCH', '/todos/3');

    xhr.setRequestHeader('content-type', 'application/json');

    xhr.send(JSON.stringify({ completed: false }));

    xhr.onload = () => {
      if ( xhr.status === 200 ){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
```

```
      }
    };
  </script>
</body>
</html>
```

### 44.3.8 DELETE 요청

```
<!DOCTYPE html>
<html>
<body>
  <pre></pre>
  <script>
    const xhr = new XMLHttpRequest();
    xhr.open('DELETE', '/todos/4');

    xhr.send();

    xhr.onload = () => {
      if ( xhr.status === 200 ){
        document.querySelector('pre').textContent = xhr.response;
      }
      else{
        console.log('Error', xhr.status, xhr.statusText);
      }
    };
  </script>
</body>
</html>
```